# Week 11: File I/O

## Leermiddelen

Heb je hulp nodig bij een onderwerp van deze week, dan bieden onderstaande links hulp.  
(klik op de tekst met ctrl+click om de links te openen)

* HFC# hoofdstuk 9 “Reading and writing files”

File i/o voorbeelden voor veelgebruikte i/o taken  
<http://msdn.microsoft.com/en-us/library/ms404278.aspx>

Reading and writing text files  
<http://msdn.microsoft.com/en-us/library/db5x7c0d.aspx>  
<http://msdn.microsoft.com/en-us/library/6ka1wd3w.aspx>  
<http://www.blackwasp.co.uk/ReadTextFiles.aspx>  
<http://www.blackwasp.co.uk/WriteTextFiles.aspx>

Reading and writing binary files  
<http://msdn.microsoft.com/en-us/library/system.io.binarywriter.aspx>  
<http://msdn.microsoft.com/en-us/library/system.io.binaryreader.aspx>  
  
(binaire) serialisatie:  
<http://msdn.microsoft.com/en-us/library/System.Runtime.Serialization.Formatters.Binary.BinaryFormatter.aspx>

## Opgave 1

Werk HFC# hoofdstuk 7 (of hoofdstuk 9 bij HCF 2nd edition) “Reading and writing files” door. Dat wil zeggen:

* Lees alles, sla niets over.
* Werk alle opgaven en quizzen uit, die in het hoofdstuk staan.

Zit je vast of heb je vragen? Vraag hulp aan de docent.

## Opgave 2

Het .NET Framework kent een aantal klassen die ervoor zorgen dat veel voorkomende bewerkingen met stations (C:, D:, etc.), mappen en bestanden gemakkelijk kunnen worden uitgevoerd. De meeste klassen bevinden zich in de namespace System.IO

In het Head First boek staan een aantal van deze mogelijkheden beschreven. Een eenvoudige klasse is **Path**. Deze bevat een aantal methoden waarmee je de padinformatie van zowel bestanden als mappen eenvoudig kunt bepalen en bewerken. De andere klassen binnen System.IO zijn te verdelen in twee typen:

- Informatieve klassen

- klassen om specifieke taken uit te voeren

De informatieve klassen **FileInfo**, **DirectoryInfo** en **DriveInfo** hebben een groot aantal eigenschappen waarmee je informatie kunt krijgen over bestanden, mappen en stations. De eerste twee klassen erven van de basisklasse **FileSystemInfo**. Hierdoor hebben ze veel eigenschappen en methoden gemeen.

Het drietal klassen om bepaalde handelingen uit te voeren zijn **Path**, **File** en **Directory**. Hiermee wordt het kopiëren, verplaatsen en verwijderen van bestanden of mappen heel eenvoudig. De methoden zijn statische methoden.

Het .Net Framework heeft uiteraard ook op het gebied van de gebruikers interface en het omgaan met mappen en bestanden het een en ander te bieden. Er is bijvoorbeeld een standaarddialoogvenster voor het openen of opslaan van bestanden. Ook is er een dialoogvenster om een map te selecteren of de voortgang van een kopieer actie weer te geven.

In deze opdracht onderzoek je verschillende mogelijkheden van System.IO

Voorbereidende vragen

* Waarvoor staat de afkorting “IO” en waarom staan de genoemde classes juist in deze namespace?
* Wat heeft het voor jou als programmeur voor gevolgen dat de methoden van **Path**, **File** en **Directory** statisch zijn?

Opdrachten

#### Opdracht 2a

Bepaal padinformatie van bestanden en mappen. Gebruik hiervoor de verschillende statische methode uit de klasse **Path**. Toon de informatie op een manier naar keuze (bijv. middels Console of in een Form)

#### Opdracht 2b

Bepaal welke stations op je systeem beschikbaar zijn. Gebruik hiervoor de methode **DriveInfo**.GetDrives() om een lijst van stations te bepalen. Kijk of een USB-stick ook wordt herkend als drive.

#### Opdracht 2c

Je wilt een bestaande map of directory naar een andere map of station kopiëren of verplaatsen. Gebruik de methode **File**.Copy() voor het kopiëren en een andere, zelf op te zoeken, methode om te verplaatsen.

#### Opdracht 2d

Maak gebruik van de mogelijkheid om een bestaande map of directory te selecteren. Dit gaat met de klasse **FolderBrowserDialog**. Maak eerst een myDocumentsFolder die je kunt meegeven aan de **FolderBrowserDialog**. De myDocumentsFolder kun je aanmaken op de volgende manier:

String myDocumentsFolder = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);

#### Opdracht 2e

Je wilt weten in welke map je programma (.exe bestand) zich bevindt. Hiervoor lees je de eigenschap **Application**.StartupPath uit of gebruik je de eigenschappen van de klasse **AppDomain**.

#### Opdracht 2f

Je wilt bepalen of een bepaald bestand bestaat en daarna dit bestand verwijderen. Gebruik hiervoor de methode Exists() en Delete() uit de klasse **File**.

#### Opdracht 2g

Op zes achtereenvolgende regels van een textfile (deze moet je zelf even aanmaken) staan de voornaam, achternaam, geslacht, adres, woonplaats en het aantal airmiles van een persoon.

Door op de knop "laad persoon info" te drukken, worden deze gegevens kenbaar gemaakt in Textboxen op het scherm. Programmeer dit en maak hierbij gebruik van een **OpenFileDialog**.

#### Opdracht 2h

En nu andersom: in de zes textboxen staan de voornaam, achternaam, geslacht, adres, woonplaats en het aantal airmiles van iemand.

Door op de knop "schrijf persoon info" te drukken, worden deze gegevens naar een textfile geschreven. De naam van deze textfile moet zijn"<voornaam><achternaam>.txt". Programmeer dit en maak hierbij gebruik van een **SaveFileDialog**.

## Opgave 3

Context

Bij een woordspel moet een gebruiker steeds een woord proberen te raden. Daartoe moet het systeem de mogelijkheid hebben om steeds een nieuw te raden woord te genereren.

Hierbij wordt gebruik gemaakt van een textfile, waarin tientallen woorden staan. In de eerste regel van de file staat het aantal woorden van die file in totaal. Op elke volgende regel van de textfile staat een aantal woorden. De woorden zijn gescheiden door spaties.

Voorbeeld van zo’n bestand (woorden.txt):

|  |
| --- |
| 10  voetbal Eindhoven Gomez  corner winnen strafschop landskampioen  topscorer  trainer nabespreking |

Merk op dat de file een regel zonder woorden bevat. Een aanroep van de methode levert dan als waarde één van de 10 woorden uit de file op.

Voorbereidende vraag

* Kan een woord in de textfile spaties bevatten als de woorden gescheiden zijn door spaties? Waarom?
* Heeft het nut om het aantal woorden in de file op te slaan op de eerste regel van de textfile? Of zou je op een andere manier achter de hoeveelheid woorden in de file kunnen komen?

#### Opdracht 3a

* Maak een methode die als returnwaarde een willekeurig woord uit de file teruggeeft.
* Zorg dat je het programma zo vaak uitvoert als nodig is om te zien dat daadwerkelijk alle woorden in het bestand af en toe worden teruggegeven en dat er geen fouten optreden.

#### Opdracht 3b

Omdat de methode vaak zal worden aangeroepen is het verstandig om niet telkens de textfile van schijf te openen maar één keer het bestand in het geheugen te laden en telkens uit de lijst van woorden in het geheugen een willekeurig woord te kiezen. Bouw deze performance optimalisatie in. Tip: maak gebruik van een Collection klasse (List?) om de woorden in het geheugen te laden, dat gaat gemakkelijker dan dat je een standaard array zou gebruiken.

## Opgave 4 Deze opgave moet je inleveren en wordt beoordeeld.

Neem als uitgangspunt je uitwerking van het dierenasiel project uit de vorige toetsopgave. De applicatie moet voorzien worden van drie extra features:

* De gebruiker moet alle dieren in de administratie op kunnen slaan in een file.
* De gebruiker moet alle dieren in de administratie kunnen vervangen door de dieren die op zijn geslagen in een file.
* In de code moeten exceptions op de juiste manier toegepast worden. Zowel in de nieuwe code als in de bestaande dierenasiel code.

Voeg daarvoor de volgende methoden toe aan de Administration klasse:

/// <summary>

/// Saves all animals to a file with the given file name using serialisation.

/// </summary>

/// <param name="fileName">The file to write to.</param>

public void Save(string fileName)

{

}

/// <summary>

/// Loads all animals from a file with the given file name using deserialisation.

/// All animals currently in the administration are removed.

/// </summary>

/// <param name="fileName">The file to read from.</param>

public void Load(string fileName)

{

}

Implementeer deze methoden en gebruik serialisatie voor het wegschrijven en laden van de gegevens. Zorg ook voor correcte exception afhandeling.

Men wil de gegevens uit de administratie ook exporteren naar andere programma’s, die niet het binaire formaat van de geserialiseerde C# files kennen. Een manier om dit te doen is een tekst file weg te schrijven. Ieder ander programma (ook niet C# programma’s) kan immers een tekst file uitlezen.

/// <summary>

/// Exports the info of all animals to a text file with the given file name.

///

/// Each line of the file contains the info about exactly one animal.

/// Each line starts with the type of animal and a colon (e.g. 'Cat:' or 'Dog:')

/// followed by the properties of the animal seperated by comma's.

/// </summary>

/// <param name="fileName">The text file to write to.</param>

public void Export(string fileName)

Implementeer deze methode. Zorg voor correcte exception afhandeling.

Maak vervolgens een nieuw Windows Forms programma “AnimalFileImporter” aan, dat deze file weer kan importeren (inlezen) en gegevens uit de eerder geëxporteerde file aan de gebruiker kan laten zien.

Het importeerprogramma moet:

* De gebruiker vragen om een naam van het te importeren bestand.
* Na het inlezen van het opgegeven bestand
  + Een listbox tonen met daarin de volgende info van alle ingelezen katten: chipnummer, naam, geboortedatum en de slechte gewoonten van de kat.
  + Een listbox tonen met daarin de volgende info van alle ingelezen honden: chipnummer, naam en geboortedatum.

Verder moeten alle bugs die aanwezig zijn in de eerder uitwerking verholpen worden en alle eerder verkregen feedback moet gebruikt zijn om het programma te verbeteren. Deze opgave is dus ook een reparatiekans voor eerder nog niet aangetoonde leerdoelen en eerder gemaakte fouten.

De criteria voor de beoordeling zijn:

* De gevraagde functionele eisen zijn geïmplementeerd.
* **Je heb ALLE feedback op je vorige uitwerkingen verwerkt in deze update van het dierenasiel programma**.
* Je hebt excepties op de juiste manier toegepast.
* Je kan het ‘hoe en waarom’ van de gemaakte code verdedigen.
* Het programma vertoont geen bugs (blijft hangen, crashed, …)
* De gemaakte code is netjes: goede encapsulatie (private/public), duidelijke variabele namen, netjes inspringen, geen dubbele code, zelf toegevoegde methoden hebben een duidelijk doel en een duidelijke naam.
* Juist gebruik van loops (for/while/foreach)
* Juist gebruik van keuze structuren (if/else/else if/switch).